IN THE CLAIMS

Pending claims follow:

1. (Previously Presented) A method for shadow mapping, comprising:
performing an offset operation to generate a depth value;
identifying a value of a slope; and
conditionally clamping the depth value based on the value of the slope.

2. (Previously Presented) The method as recited in claim 1, wherein the shadow mapping process includes rendering a primitive from a light space perspective.

3. (Original) The method as recited in claim 1, wherein the depth value is clamped if the value of the slope is greater than a predetermined amount.

4. (Previously Presented) The method as recited in claim 1, wherein the clamping includes the steps of: identifying vertex depth values of vertices of a primitive; comparing at least one of the vertex depth values with the depth value generated by the offset operation; and clamping the depth value generated by the offset operation based on the comparison.

5. (Previously Presented) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is less than a maximum vertex depth value.

6. (Previously Presented) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to a greatest vertex depth value if the greatest vertex depth value is less than the depth value generated by the offset operation.

7. (Previously Presented) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is greater than a least one of the vertex depth values.

8. (Previously Presented) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the least one of the vertex depth values if a least one of the vertex depth values is greater than the depth value generated by the offset operation.

9. (Original) The method as recited in claim 1, wherein the offset operation includes a polygon offset operation in accordance with the OpenGL® programming language.

10. (Previously Presented) A computer program embodied on a computer readable medium for shadow mapping, comprising:
a code segment for performing an offset operation to generate a depth value;
a code segment for identifying a value of a slope; and
a code segment for conditionally clamping the depth value based on the value of the slope.

11. (Original) The computer program as recited in claim 10, wherein the depth value is clamped if the value of the slope is greater than a predetermined amount.

12. (Previously Presented) The computer program as recited in claim 10, wherein the clamping includes: identifying vertex depth values of vertices of a primitive; comparing at least one of the vertex depth values with the depth value generated by the offset operation; and clamping the depth value generated by the offset operation based on the comparison.

13. (Previously Presented) The computer program as recited in claim 12, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is less than a maximum vertex depth value, and wherein the depth value generated by the offset operation is clamped to the greatest vertex depth value if the greatest vertex depth value is less than the depth value generated by the offset operation.

14. (Previously Presented) The computer program as recited in claim 12, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is greater than a least one of the vertex depth values, and wherein the depth value generated by the offset operation is clamped to the least one of the vertex depth values if the least one of the vertex depth values is greater than the depth value generated by the offset operation.

15. (Previously Presented) A system for shadow mapping, comprising:
logic for performing an offset operation to generate a depth value;
logic for calculating and identifying a value of a slope; and
logic for conditionally clamping the depth value based on the value of the slope.

16. – 28. (Cancelled)

29. (Previously Presented) A method for performing shading calculations in a graphics pipeline, comprising:
performing a first shading calculation in order to generate output utilizing a single shader unit of a graphics pipeline;
saving the output; and
performing a second shading calculation using the output in order to generate further output utilizing the single shader unit of the graphics pipeline.

30. (Previously Presented) The method as recited in claim 29, wherein the first shading calculation includes [(1-s)*(Color_diff + Color_spec)] for generating an output A, and the second shading calculation includes [Color_amb + A], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, and Color_amb is an ambient color variable.

31. (Previously Presented) The method as recited in claim 29, wherein the first shading calculation includes [((1-s)* Color_diff) + Color_amb] for generating an output A, and the second shading calculation includes [A*Texture_det + (1-s)* Color_spec], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, Color_amb is an ambient color variable, and Texture_det is a detail texture variable.

32. (Previously Presented) The method as recited in claim 29, wherein the first and second shading calculations together include a diffuse color variable, a specular color variable, and an ambient color variable.

33. (Previously Presented) The method as recited in claim 32, wherein the variables are decoupled.

34. (Previously Presented) The method as recited in claim 29, wherein the method is carried out with a system comprising:

(a) a shading module for performing the first shading calculation in order to generate the output;

(b) a texture look-up module coupled to the shading module for retrieving texture information using texture coordinates associated with the output;

(c) a feedback loop coupled between an input and an output of the shading module for performing the second shading calculation using the texture information from the texture look-up module in order to generate further output; and

(d) a combiner module coupled to the output of the shading module for combining the output generated by the shading module.

35. (Previously Presented) A computer program embodied on a computer readable medium for performing shading calculations in a graphics pipeline, comprising:

a code segment for performing a first shading calculation in order to generate output utilizing a single shader unit of a graphics pipeline;

a code segment for saving the output; and

a code segment for performing a second shading calculation using the output in order to generate further output utilizing the single shader unit of the graphics pipeline.

36. (Previously Presented) The computer program as recited in claim 35, wherein the first shading calculation includes [(1-s)*(Color_diff + Color_spec)] for generating an output A, and the second shading calculation includes [Color_amb + A], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, and Color_amb is an ambient color variable.

37. (Previously Presented) The computer program as recited in claim 35, wherein the first shading calculation includes [((1-s)* Color_diff) + Color_amb] for generating an output A, and the second shading calculation includes [A*Texture_det + (1-s)* Color_spec], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, Color_amb is an ambient color variable, and Texture_det is a texture detail variable.

38. (Previously Presented) The computer program as recited in claim 35, wherein the first and second shading calculations together include a diffuse color variable, a specular color variable, and an ambient color variable.

39. (Previously Presented) The computer program as recited in claim 38, wherein the variables are decoupled.

40. (Previously Presented) The computer program as recited in claim 35, wherein the code segments are carried out with a system comprising:

(a) a shading module for performing the first shading calculation in order to generate the output;

(b) a texture look-up module coupled to the shading module for retrieving texture information using texture coordinates associated with the output;

(c) a feedback loop coupled between an input and an output of the shading module for performing the second shading calculation using the texture information from the texture look-up module in order to generate further output; and

(d) a combiner module coupled to the output of the shading module for combining the output generated by the shading module.

41. (Previously Presented) A system for performing shading calculations in a graphics pipeline, comprising:
logic for performing a first shading calculation in order to generate output utilizing a single shader unit of a graphics pipeline;
logic for saving the output; and
logic for performing a second shading calculation using the output in order to generate further output utilizing the single shader unit of the graphics pipeline.

42. (Previously Presented) The method as recited in claim 29, wherein the shading calculations involve shadow modulation.

43. (Previously Presented) The method as recited in claim 42, wherein the shadow modulation involves more than one function.

44. (Previously Presented) The method as recited in claim 29, wherein the first and second shading calculations involve decoupled variables.